PROYECTO FINAL DE CARRERA

DISEÑO E IMPLEMENTACIÓN DE UN SOFTWARE GRAFICADOR DE TABLATURAS DE BAJO ELÉCTRICO ESTANDAR EN TIEMPO REAL

(design and implementation of a software plotter of tabs for standard electric bass in real time)

Titulación:

Ingeniería en sistemas y computación

Autor:

Alejandro Plaza Parra

Directora:

ING. Ivonne Castaño Osorio

Universidad Tecnológica de Pereira

Pereira, Risaralda, Colombia

Julio, 2019

Abstract

This project explains the design and implementation of a software that makes the graph of the tablature of notes entered by means of a standard electrical bass in real time. For this purpose, you must obtain the value of the frequency in the notes entered. Frequency estimation is performed using the Fast Fourier Transform, processing the sound of the notes in real time and comparing it to a fixed threshold to determine the position of the note on the electric bass mast. The data obtained within this process of decomposition and comparison are used as parameters so that the software can graph the respective simplified tablature, showing on screen the string and fret where that note was generated. These comparison frequencies were previously found by the same decomposition method, since these values should not vary if the instrument is tuned. The results show a good approximation in the graphical representation of entered melodies, being able to correctly generate 75% of the positions of these.

Resumen

Este proyecto explica el diseño e implementación de un software que realice la gráfica de la tablatura de notas ingresadas mediante un bajo eléctrico estándar en tiempo real. Para este propósito, se debe obtener el valor de la frecuencia en las notas ingresadas. La estimación de la frecuencia se realiza mediante el uso de la transformada rápida de Fourier, procesando el sonido de las notas en tiempo real y comparándola con un umbral fijo para determinar la posición de la nota en el mástil del bajo eléctrico. Los datos obtenidos dentro de este proceso de descomposición y comparación son utilizados como parámetros para que el software pueda graficar la respectiva tablatura simplificada, mostrando en pantalla la cuerda y el traste en donde esa nota fue generada. Estas frecuencias de comparación fueron halladas previamente mediante el mismo método de descomposición, dado que estos valores no deben variar si el instrumento esta afinado. Los resultados muestran una buena aproximación en la representación gráfica de melodías ingresadas, pudiendo generar correctamente el 75% de las posiciones de estas.

Agradecimientos

Agradezco a muchas personas que este trabajo haya podido ser realizado con éxito es probable que se me olvide mencionar a alguien, así que pido disculpas de antemano.

En primer lugar, quiero agradecerles a esos maestros que me impartieron clases durante estos años, generando en mi de manera correcta el interés por la carrera que necesitaba y me impulso a terminarla.

Le agradezco a Ivonne Castaño su sabiduría y sus consejos en cada momento de la evolución de este proyecto, la orientación, todas las aportaciones en mi crecimiento tanto personal como profesional y en la confianza depositada en mí.

Agradezco a Diego Fernando Hincapié por su colaboración continua en la implementación técnica del proyecto, por no abandonarme y estar pendiente de mis avances.

Agradezco a mis compañeros de carrera, especialmente a Natalia Franco, Fernán Cañas, Juan David López, Sebastián Sánchez, Edgar Martínez por compartir conmigo esta etapa de mi vida, por brindarme su amistad y apoyo, además de los momentos inolvidables que vivimos.

Agradezco a mis amigos, especialmente a Mateo Bedoya, Sergio Villaquiran y José Santa quienes con sus conocimientos sobre la música me instruyeron y ayudaron a generar un proyecto basado en este arte, por los buenos momentos y experiencias, y por aligerar mis malos momentos con su invaluable ayuda y consejos.

Agradezco a Nicolás Ruiz por sus conocimientos, ayuda y apoyo a través de la carrera, por orientarme en la dirección correcta al resolver problemas lógicos de programación.

También quiero agradecer a Carlos Augusto Meneses y a Marta Lucy Estrella por su disposición, orientación y ayuda en las etapas finales de la carrera

A mis padres Hernando Plaza y Lorena Parra porque sin ellos no estaría realizando mi sueño, por el esfuerzo, apoyo y confianza depositados en mí.

A mi novia Karol Dahiana por brindarme el apoyo y consuelo necesario en épocas debilidad, por todos sus ánimos y palabras y su presencia continua.

Índice

Abstract **ii**

Resumen **iii**

Agradecimientos **iv**

Índice **vi**

Índice de Tablas **ix**

Índice de Figuras **x**

Introducción **1**

1. Definición del problema **2**

2. Objetivos **3**

2.1 Objetivo general 3

2.2 Objetivos específicos 3

3. Justificación **4**

4. Metodología **5**

4.1 Metodología del desarrollo 5

4.1.1 Modelo en cascada 5

4.1.2 Requisitos del sistema 6

4.1.3 Modelamiento del sistema 7

4.1.3.1 Diagrama de casos de uso 7

4.1.3.2 Diagrama de clases 7

4.1.3.3 Diagrama de actividades 8

4.1.3.4 Diagramas de clases de diseño 8

4.1.3.4.1 Diseño Modelo- Vista- Controlador 8

4.1.3.4.2 Comunicación entre campos 8

4.1.3.5 Diagrama Entidad-Relación 9
4.1.3.6 Diagrama de despliegue 9
4.1.4 Entorno de desarrollo 10
4.1.4.1 Python 10
4.1.5 Principios matemáticos 10
4.1.5.1 Transformada rápida de Fourier 10
4.1.5.2 Rangos 10
5 Limitaciones del proyecto 11
5.1 Timbre 11
6 Marco Teórico 13
6.1 Sonido 13
6.1.1 Ruido 14
6.1.2 Velocidad de propagación 14
6.1.3 Partes de la onda sonora 14
6.2 Transformada de Fourier 15
6.2.1 Propiedades de la transformada de Fourier 17
6.3 Herramientas actuales 18
7 Desarrollo 19
7.1 Análisis 19
7.1.1 Catálogo de requisitos 19
7.1.1.1 Requisitos funcionales 20
7.1.1.1.1 Uso del software 20
7.1.1.1.2 Sistema 20
7.1.1.2 Requisitos no funcionales 22
7.1.2 Modelado 23
7.1.2.1 Diagrama de casos de uso 24
7.1.2.2 Diagrama de clases 26
7.1.2.3 Diagrama Entidad-Relación 27
7.2 Diseño 28
7.2.1 Arquitectura del software 28
7.2.2 Diagrama de despliegue 29

7.2.3 Diagramas de actividades 30
7.3 Tecnologías utilizadas 33
7.4 Diseño interfaz de usuario 34
7.5 Implementación 35
7.5.1 Desarrollo del proyecto 35
7.5.2 Pruebas 41
8 Conclusiones y dificultades 44
9 Recomendaciones 46
Referencias 47
Anexo A 48
Anexo B 50

Índice de Tablas

7.1 Casos de uso 26
7.2 Librerías necesarias dentro de python 29
7.3 Prueba de Requisito funcional 1 41
7.4 Prueba de Requisito funcional 5 42
7.5 Prueba de Requisito funcional 7 42
7.6 Prueba de Requisito funcional 8 43

Índice de Figuras

4.1 Etapas del modelo en cascada 6

6.1 Componentes de una onda sonora 15

6.2 Valor eficaz o R.M.S 15

6.3 Una señal genérica se transforma por una sumatoria de señales sinusoidales 16

6.4 Transformada de Fourier 17

6.5 Transformada rápida de Fourier 17

7.1 Diagrama de casos de uso 24

7.2 Diagrama de clases 27

7.3 Diagrama Entidad-Relación 27

7.4 Modelo – Vista – Controlador 28

7.5 Diagrama de despliegue 29

7.6 Diagrama de actividad, caso de uso PassiveListen 30

7.7 Diagrama de actividad, caso de uso Comparar 31

7.8 Diagrama de actividad, caso de uso Graficar 32

7.9 Diagrama de actividad, caso de uso GraficarL 33

7.10 Mockup inicio 34

7.11 Mockups para el graficado 34

7.12 Clases 35

7.13 Método Rms 36

7.14 Método Passivelisten y cargar 37

7.15 Método comparar y rangos de frecuencia 38

7.16 Método Graficar 39

7.17 Método Grabar y Load 40

7.18 Método Llenar_lista 40

7.19 Método GraficarL 41

A1 Pantalla de inicio 48

A2 Ventana al presionar “Graficar” 48

A3 Resultado “Graficar” 48

A4 Re visualizar 49

Introducción

El objetivo del presente proyecto es crear un intérprete graficador que permita generar y guardar tablature de manera autónoma y digital. Este desarrollo permitirá analizar los sonidos de las notas emitidas por el bajo eléctrico para descomponerlos en frecuencias y poder determinar en qué posición o traste fue generada y en qué cuerda.

Para poder llevar a cabo el desarrollo del software se debe tener en cuenta que el mercado cuenta con múltiples tipos y marcas de bajos eléctricos que varían tanto en número de cuerdas como en la tonalidad que manejan, los bajos eléctricos que se usan más comúnmente son aquellos denominados estándar, ya que permiten el dinamismo que se necesita si se está empezando en el mundo musical. Estos serán el enfoque del proyecto ya que, al tener un mayor grado de aceptación, permitirá que el software tenga una mayor expansión de uso.
Además, integra una interfaz gráfica amigable con el usuario, permitiendo su uso intuitivo acertado.
El bajo eléctrico estándar cuenta con cuatro cuerdas y afinación Sol, Re, La, Mi.

1

Definición del problema

Al componer música, el método tradicional consiste en escribir las notas a lo largo de un conjunto de líneas llamado pentagrama, que contiene armaduras o claves que definen que cinco notas aparecerán sobre el conjunto (sol, fa, do) donde cada línea representa una nota (do, re, mi, fa, sol, la, sí), y sobre la cual se escriben ciertos símbolos que representan la tonalidad y su duración (♩,♪,♫,♬….). Esto funciona bastante bien y es consistente, pero sólo para quien conoce esta nomenclatura y ya tiene experiencia.

Las herramientas de software actuales presentan soluciones con respecto a la digitalización de las tablaturas, pero no a la automatización de estas. es decir, poder componer la música y obtener las tablaturas a partir del software, lo cual permitiría la ejecución eficaz de este proceso y se le facilitaría el trabajo a los artistas más inexpertos en el tema.

Otro aspecto importante es la posibilidad de acceder a un registro histórico de composiciones que hayan gustado para poder ser revisadas nuevamente pasado un tiempo, evitando que estas se deterioren o se pierdan.

2

Objetivos

2.1 Objetivo general

El objetivo general de este proyecto es el diseño e implementación de un software que realice la gráfica de la tablatura de notas generadas mediante un bajo eléctrico en tiempo real; el cuál se llevará a cabo mediante los diferentes objetivos específicos detallados a continuación.

2.2 Objetivos específicos

Los objetivos específicos para lograr el objetivo general son los siguientes:

- Elaborar el plan de proyecto
- Levantar requerimientos.
- Definir los rangos de frecuencia de las notas del bajo eléctrico.
- Realizar el análisis del sistema.
- Realizar el diseño del sistema.
- Implementar el proyecto.
- Realizar las pruebas

3

Justificación

En la actualidad los bajistas llevan registro de su progreso de la manera tradicional o confiando sólo en su memoria, lo que ocasiona que se pierda precisión a la hora de reproducir la composición original y por tanto, se pierde tiempo ajustando de nuevo la melodía. ¿De qué manera podría un bajista automatizar la composición de su música para garantizar la integridad?

De igual manera aunque estos registros sean llevados en una bitacora, no tienen la posibilidad de mantener al dia sus progresos, ya que muchas de estas obras con el tiempo tienden a deteriorarse ocacionando perdida de esfuerzo y tiempo.

4

Metodología

Para desarrollar el proyecto, que es de tipo de aplicación tecnológica, se llevará a cabo el ciclo de desarrollo de software, utilizando un modelo en cascada. Se definirán los requisitos funcionales y no funcionales, se hará el respectivo análisis y diseño utilizando el lenguaje de modelado UML y posteriormente se realizará la implementación utilizando lenguaje python, ya que este permite el manejo de archivos multimedia e interfaz gráfica de manera más precisa que en otros lenguajes. Cada uno de los apartados mencionados serán explicados parcialmente en este capítulo

4.1 Metodología del desarrollo

4.1.1 Modelo en cascada

El modelo en cascada es quien nos indica que el proceso de desarrollo es secuencial, describiendo este último como un conjunto de etapas que se ejecutan una tras otra. Se representan una encima de la otra y siguiendo el flujo de ejecución de arriba hacia abajo, como una cascada.

Cada fase hace parte del ciclo de desarrollo del software y nos da más información sobre la dirección del proyecto. Al ser un modelo secuencial las etapas principales son de suma importancia para las etapas posteriores, ya que estas son las bases del proyecto

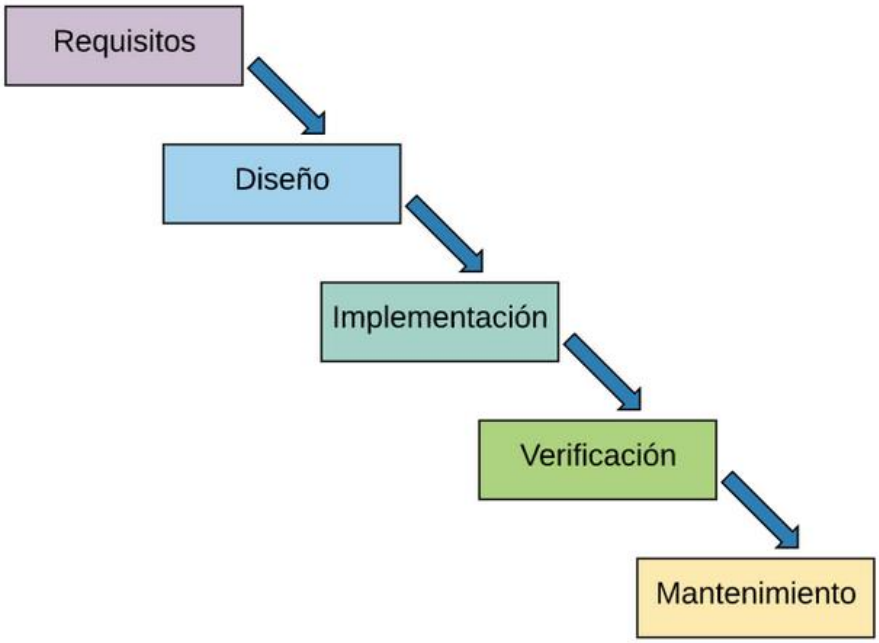

Figura 4.1: Etapas del modelo en cascada.

4.1.2 Requisitos del sistema

Un requisito es definido como una necesidad documentada sobre el contenido, funcionalidad de un producto y servicio, estos requisitos son levantados al inicio de cualquier proyecto de aplicación ya que a su vez delimitan el alcance del proyecto y se especifica cómo es que se desea que funcione. Estos se dividen principalmente en dos.

Los **Requisitos funcionales** pueden ser una descripción detallada de lo que un sistema debe hacer, o está en capacidad de hacer, se puede dividir en apartados si es posible, los requisitos específicos con los que puede interactuar el usuario (Interfaz, botones, etc.) y

los requisitos sobre la funcionalidad propios del sistema, lo que debe realizar internamente sin que el usuario se entere.

Los **Requisitos no funcionales** son especificaciones únicamente propias del sistema, no referentes directamente a funcionalidades, pero si a sus características de funcionamiento, estos pueden ser rendimiento, calidad, disponibilidad.

4.1.3 Modelamiento del sistema

Cada metodología de desarrollo tiene su propio enfoque, entre ellos el más utilizado en el modelamiento de diseño de software es el lenguaje de modelado UML el cual es un lenguaje visual común, semántico y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas. En general los diagramas UML describen los limites, la estructura y el comportamiento de los objetos que contiene el sistema. Entre los más conocidos (y que se implementaron en el proyecto) están:

4.1.3.1 Diagrama de casos de uso

El diagrama de casos de uso representa la forma en como un usuario (actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan para cumplir con las funcionalidades del sistema. A continuación, el diagrama para el proyecto propuesto.

4.1.3.2 Diagrama de clases

El diagrama de clases es un tipo de diagrama de estructura estática que describe la estructura de un sistema, mostrando las clases del sistema, sus atributos, operaciones o métodos, y las relaciones entre los objetos. De esta manera, a continuación, veremos el diagrama de clases del sistema.

4.1.3.3 Diagrama de actividades

El diagrama de actividad es un diagrama de flujo de proceso multipropósito que se una para modelar el comportamiento del sistema, estos se pueden usar para modelar un caso de uso en particular, una clase, o un método complicado. Por ello y debido a que el sistema cuenta con diferentes casos críticos se han realizado los siguientes modelos.

4.1.3.4 Diagrama de clases de diseño

La arquitectura del sistema tendrá el formato Modelo – Vista – Controlador constituyendo las funcionalidades de cada uno de estos componentes.

4.1.3.4.1 Diseño Modelo- Vista- Controlador

El diseño Modelo – Vista – Controlador, o también llamado diseño MVC, está orientado a objetos dentro de la programación del software. Este permite separar la programación del sistema en tres campos. El Modelo contiene información del sistema, como pueden ser: las variables, las especificaciones, etc. El Controlador se encarga de como presentar lo que se encuentra en el modelo al usuario en una interfaz amigable e intuitiva. Finalmente, la Vista es la encargada de enseñar lo que le ordene el controlador.

4.1.3.4.2 Comunicación entre campos

La comunicación entre los tres campos no es absoluta. El Controlador puede comunicarse con el Método sin ningún problema, y puede obtener toda la información que necesite cuando lo requiera, ya que el Controlador es el encargado de poner en pantalla el Modelo. También el Controlador puede comunicarse con la Vista para poder gestionar la interfaz de usuario. Mientras que la comunicación entre el Modelo y la Vista, idealmente no se debería contemplar, debido a que el Modelo es independiente de la interfaz de usuario. También se podría realizar un Modelo que se

adaptara a la vista, pero esto afectaría a la reusabilidad del código y haría más complicada la búsqueda de errores durante la depuración.

Además, la comunicación entre Vista y Controlador se puede realizar de tres maneras. Una forma es que la Vista a través de acciones pueda comunicarle al Controlador lo que el usuario está realizando en la interfaz. La segunda forma es que el Controlador tenga el control de lo que pase en la vista través de delegados. Por último, análogo a la anterior, el Controlador gestiona la información que quiera mostrar la vista a través de una fuente de datos. La comunicación entre el Modelo y el Controlador noes directa. Cuando el Modelo necesita actualizar información debe utilizar notificaciones, para notificarle al Controlador los nuevos cambios.

4.1.3.5 Diagrama Entidad-Relación

El diagrama entidad relación nos enseña cómo están relacionadas las entidades dentro del sistema, son usados para diseñar o depurar bases de datos relacionales si así se requiere. A continuación, veremos el diagrama Entidad-Relación del sistema.

4.1.3.6 Diagrama de despliegue

Este es quien modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra como los elementos y artefactos del software se trazan sobre estos nodos. A continuación, veremos el diagrama de despliegue del proyecto.

4.1.4 Entorno de desarrollo

4.1.4.1 Python

Python es un lenguaje de programación interpretado cuya principal característica es proporcionar una sintaxis que favorezca un código legible. Es a su vez un lenguaje multiparadigma, ya que soporta programación orientada a objetos, programación imperativa y programación funcional. Cuenta con licencia de código abierto lo que permite la libre utilización y distribución del mismo

4.1.5 Principios matemáticos

4.1.5.1 Transformada Rápida de Fourier

Es una herramienta fundamental en el procesado de señales digital. Esta se trata de un algoritmo para el cálculo de la Transformada Discreta de Fourier, su importancia se debe a que elimina una gran parte de los cálculos repetitivos a que está sometida la DFT, reduciendo errores de redondeo. Su implementación por otra parte requiere de programas que puedan hacer uso de un microprocesador específico para este tipo de operaciones (DSP)

4.1.5.2 Rangos

El rango o recorrido es el intervalo entre el valor mínimo y el valor máximo, es usado comúnmente para definir momentos de decisión dentro de los sistemas, creando a su vez condiciones, es bastante útil cuando el sistema se basa en operaciones con números.

5

Limitaciones del proyecto

5.1 Timbre

Los bajos eléctricos se componen por varios elementos, los cuales definen su diseño y su timbre al tocar, uno de los componentes que alteran el sonido es el calibre de las cuerdas que se usan, ya que estas de entrada modifican la nota que producen dependiendo de su entorchado y el material del que estén echas, los bajos de cuatro cuerdas de serie vienen con las afinaciones estándar (Sol, Re, La, Mi) y con cuerdas de entorchado semiliso, permitiendo versatilidad.

Las pastillas, que funcionan captando las vibraciones de las cuerdas y transmitiéndolas al amplificador, también alteran el timbre, ya que dependiendo de la distribución de las mismas captaran en mayor o menor medida estas vibraciones, los bajos estándar se componen con al menos 2 de estas pastillas cubriendo las cuatro cuerdas del mismo.

El alma del bajo eléctrico, es una varilla delgada de acero que se encuentra entre el diapasón del bajo y el mástil, se emplea para modificar la curvatura de este mástil, que es donde se encuentran las cuerdas, modificar la curvatura ocasionara que el instrumento cambie su timbre, existiendo un estándar de medición que indica la curvatura optima del instrumento.

Existen a su vez diferentes maneras de tocar un bajo eléctrico, que alteran el timbre de la nota siendo la más popular el “fingerstyle” que consiste en pulsar las cuerdas con los dedos índice y

corazón de la mano derecha, pero no es la única su variación más cercana es pulsar las cuerdas con ayuda de un “pick” o “púa”, y la variación más agresiva usada es el “slap”, que consiste en golpear directamente la cuerda con el pulgar, cada técnica genera distorsión adicional en las notas, por lo que las pastillas no captan de manera muy clara las vibraciones, ocasionando un timbre distante entre cada técnica.

Al existir tantos factores que podrían alterar el funcionamiento del software se ha decidido delimitar bastante el alcance del proyecto, definiendo entonces que se centrara en un bajo con cuatro cuerdas (Sol, Re, La, Mi), con entorche semiliso, que contengan mínimo 2 pastillas completas, que su alma esté debidamente acomodada y que la técnica utilizada sea “figerStyle”.

Se ha utilizado entonces un bajo eléctrico marca Fander de 5 cuerdas, con alma estándar y pastillas completas como dedicadas. Aunque este ejemplar posea una cuerda demás para cumplir las especificaciones no tendrá inconvenientes para ajustarse a lo requerido, ya que la cuerda numero 5 (afinación Si) será ignorada. Como acompañante se ha utilizado un amplificador Boston de 115 watts y un parlante Bose de fácil transporte.

6

Marco Teórico

En este capítulo se dan a conocer los trabajos y publicaciones previos relacionados con los objetivos de este proyecto. En la sección 6.1 se mencionan trabajos relacionados al manejo de sonidos, y en la sección 6.2 se muestran investigaciones sobre herramientas actuales que usan este mismo principio.

6.1 Sonido

El sonido audible consiste en ondas sonoras y ondas acústicas que se producen cuando las oscilaciones de la presión de aire, son convertidas en ondas mecánicas en el oído y percibidas por el cerebro, las cuales pueden viajar a través de cualquier medio sólido, liquido o gaseoso.

En la definición del sonido también debe considerarse tanto el fenómeno físico como el sicoacústico, ya que bajo la ausencia de un oyente puede existir un evento sonoro, pero no el evento auditivo.

El sonido se compone de varios componentes como el tono, el cual es determinado por la frecuencia fundamental de las ondas sonoras medidas en ciclos, este es quien nos permite distinguir entre sonidos graves y agudos. La duración, que no es más que el tiempo de propagación de la onda. El timbre acompaña a la frecuencia y es el atributo que nos permite diferenciar entre dos sonidos con igual frecuencia [1]

6.1.1 Ruido

Al igual que definimos en el apartado anterior el ruido es físicamente definible de la misma manera, solo que, acostumbramos llamar ruido a aquellos sonidos que no son de nuestro agrado, por ello la definición más aceptada internacionalmente es la que lo categoriza como un sonido no deseado, así pues, podríamos decir que, sin importar la procedencia del sonido, la diferencia entre ruido y un sonido es circunstancial. [4].

6.1.2 Velocidad de propagación

La onda sonora requiere un medio para propagarse, sin importar este medio las características no varían, estas son, su temperatura, humedad densidad y elasticidad, dependiendo de estos elementos varia la velocidad de propagación de la misma. [4]

Para este proyecto nos concierne primordialmente la propagación de las ondas sonoras por el aire, en este medio el sonido viaja a una velocidad de aproximadamente 340 m/s. Debido a que en este medio las partículas que transportan la onda chocan entre ellas, facilitan la propagación de la misma

6.1.3 Partes de la onda sonora

Las ondas sonoras en su estructura cuentan con varios componentes a tener en cuenta, sobre todo si vamos a hacer tratamiento y procedimientos con ella. Estas son en su mayoría ondas sinusoidales y cuentan con.

Longitud de onda Es la longitud de un ciclo completa de la onda (comprendiendo valle y cresta)

Periodo Es la duración en segundos de un ciclo completo de la onda

Frecuencia Es el número de ciclos u oscilaciones que se repiten en un segundo. Sus unidades son Hertzios (Hz), esta es la que musicalmente llamamos el tono, las frecuencias altas son agudas mientras que las bajas son más graves.

Amplitud Es la presión sonora o fuerza por unidad de superficie de las partículas del medio en un punto dado, se mide en pascales (Pa) o N/m^2

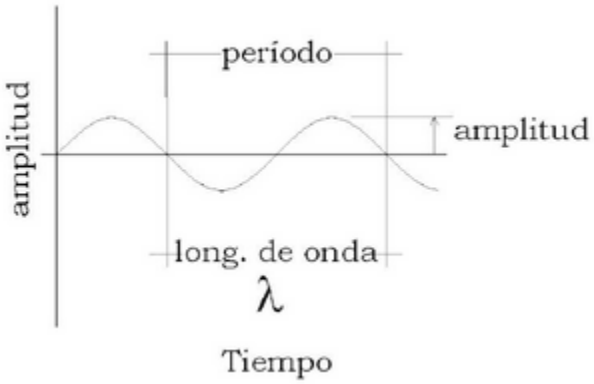

Figura 6.1 Componentes de una onda sonora

Valor Pico Es el punto máximo de presión sonora en una onda

Valor eficaz o r.m.s Es la onda sinusoidal cuya energía transportada es equivalente a la de una señal directa constante. Este es el valor más empleado y al cual nos referimos siempre que no se especifique de otro modo. Es el más representativo de las características generales de un sonido en el tiempo.

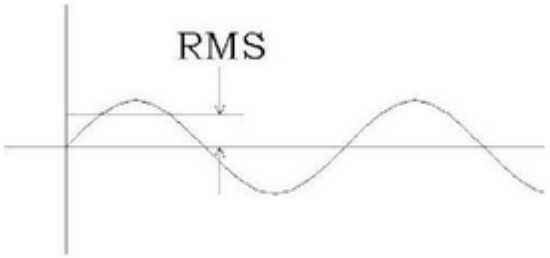

Figura 6.2 Valor eficaz o R.M.S

6.2 Transformada de Fourier

La transformada de Fourier es una herramienta de análisis utilizada en el campo científico (acústica, procesamiento de señales, radar, electromagnetismos, comunicaciones, etc.) la cual transforma una señal representada en el dominio del tiempo al dominio de la frecuencia, pero sin alterar su contenido de información, ya que solo es una forma diferente de representarla.

Su potencia radica en que nos permite descomponer una señal compleja en un conjunto de componentes de frecuencia única; sin embargo, no nos indica el instante en que han ocurrido. Por ello, esta descomposición es útil para señales estacionarias, estos componentes de la frecuencia que forman la señal compleja no cambian a lo largo del tiempo.

La transformada de Fourier debe aplicarse de – 00 a + 00 para tomar tramos debemos multiplicar la señal por una ventana temporal que nos aislé la parte requerida. Este hecho nos provoca una distorsión en el espectro obtenido, ya que el resultado es la convulsión de la transformada de la señal

Los autores proponen la utilización de la transformada de Fourier para entender los sonidos y la música de una manera más matemática ya que esta expresa que toda función periódica puede expresarse como la suma infinita de funciones seno o coseno que son múltiplos enteros n de la frecuencia, hallando su frecuencia fundamental, y a su vez cada término de seno y coseno se le conocen como armónicos.

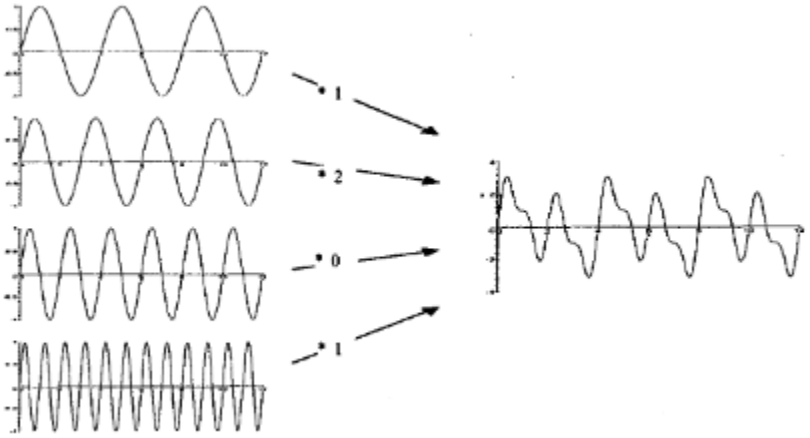

Figura 6.3: Una señal genérica se transforma por una sumatoria de señales sinusoidales

Logrando condensar su algoritmo se logra que realice los mismos cálculos, pero en un menor tiempo, dando lugar a la transformada rápida de Fourier. Estos infieren pues que la forma matemática más sencilla para la síntesis auditiva es utilizar las series de Fourier, pudiendo hallar la frecuencia del sonido basándose solo en su amplitud y su fase.

$$f(t) = c_0 + \sum_{n=1}^{\infty} \left[c_n \, e^{(n\,\omega_0 t)j} + (c_{-n} \, e^{-(n\,\omega_0 t)j} \right]$$

Figura 6.4: Transformada de Fourier

$$x[k] = \sum_{n=0}^{N-1} x[n] \, e^{-(2\pi \, n \, k/N)j}$$

Figura 6.5: Transformada rápida de Fourier.

6.2.1 Propiedades de la transformada de Fourier

Linealidad. Si las funciones x(t) y y(t) tienen como transformada a X(f) y Y(f), respectivamente, entonces las sumas de ambas tienen como transformada a X(f)+ Y(f).

Simetría. Si X(f) es la transformada de x(t), la transformada de X(t) es x(-f).

Escalado en el tiempo y en la frecuencia. Si realizamos un escalado de la variable t mediante la constante k la transformada da: $\frac{1}{|k|} x(\frac{t}{k}) \Leftrightarrow X(kf)$.

igualmente, si realizamos un escalado de la frecuencia mediante la constate k la transformada inversa nos da $x(kt) \Leftrightarrow \frac{1}{k} X\left(\frac{f}{k}\right)$.

Desplazamiento en tiempo y en frecuencia. Si desplazamos el tiempo según la constante t0, la trasformada nos queda $X(t-t_0) \Leftrightarrow x(f)e^{-j2\pi f t_0}$. Si desplazamos la frecuencia con la Constante f0, la transformada inversa nos queda $x(t)e^{-j2\pi f_0} \Leftrightarrow X(f-f_0)$.

Multiplicación frente a convolución. Definiendo la operación de convolución entre dos funciones como: $x(t) * y(t) = \int_{-\infty}^{\infty} x(\tau) y(t-\tau) d\tau$ se cumple que si multiplicamos dos funciones en el tiempo y calculamos la transformada equivale a ralizar la operación de

covolución(*: operador de convolución) entre las transformadas de ambas funciones; la operación de convolución en el tiempo equivale a la multiplicacion de frecuencias [3].

$$x(t)y(t) \Leftrightarrow X(f) * Y(f)$$
$$x(t) * y(t) \Leftrightarrow X(f)Y(f)$$

6.3 Herramientas actuales

En la actualidad el uso y tratamiento de señales tiene bastantes usos prácticos.

Algunos autores hacen mención de que el estudio de la síntesis de los sonidos de los instrumentos musicales es un tema motivador que interesa a los estudiantes de ciencias básicas, ingenierías y música ya que entrega una aproximación al tema desde diferentes perspectivas ya que el modelamiento de transmisión de ondas puede ser abordado desde la parte física y matemática a la vez, permitiendo versatilidad de aprendizaje sobre estos temas [2].

Algunos abordan este tema basándose en el funcionamiento de los afinadores actuales ya que el funcionamiento de estos dispositivos funciona de manera similar a lo planteado anteriormente, basados en que los sonidos de las cuerdas al aire de los instrumentos de cuerda tienen su propia frecuencia fundamental, la cual comparándola con registros anteriormente diseñados puede comunicarle al usuario cuando su instrumento esta afinado.

Usando este principio de tratamiento de señales y con algoritmos de recomposición y tratamiento de datos es como google implementa su búsqueda por voz implantado en nuestros teléfonos Smartphone, ya que utilizan la descomposición de los sonidos en ondas más manejables para compararlas con su inmensa base de datos, una red neuronal es la que se encarga de encontrar las similitudes para entender que es lo que se le está solicitando que busque, por lo que en ultimas somos los usuarios quienes no encargamos de entrenar a la red para que funcione más eficientemente.

En el campo de la robótica las utilizaciones de descomposición de sonidos en ondas manejables permiten que en la actualidad el androide llamado sophia pueda entender y dialogar

con personas de su entorno, realizando tanto síntesis de lo que escucha como su proceso inverso, pudiendo así hacer uso de señales computacionales para comunicarse eficazmente.

7

Desarrollo

En el presente capitulo veremos y analizaremos en profundidad las etapas criticas del proyecto, es aquí donde se plasma la fase de análisis, diseño e implementación con sus respectivas pruebas por lo que el capítulo tratara de ser lo mayormente explicativo posible.

7.1 Análisis

La fase de análisis es la parte inicial de cualquier proyecto de software, donde se definen los requisitos y se enseña una vista global de la arquitectura que se desea implementar en el sistema. Ya que en el siguiente apartado se tendrá en cuenta lo aquí plasmado como base para el diseño de todos los aspectos del software. Por ello que la fase de análisis de se suma importancia para el futuro del proyecto. Aquí es donde se sentarán las bases del proyecto.

7.1.1 Catálogo de requisitos

El catálogo de requisitos es la especificación del comportamiento que se espera de cualquier proyecto de software, estudiando las necesidades de los usuarios se ha predefinido una serie de requisitos que se consideran indispensables para el proyecto. A continuación, se muestra una enumeración y breve descripción de los requisitos establecidos para el diseño y desarrollo de la aplicación.

7.1.1.1 Requisitos funcionales

Los requisitos funcionales describen todas las interacciones que tendrán los usuarios con el software.

7.1.1.1.1 Uso del software

RF1: Graficar

1. El software deberá iniciar la operación únicamente cuando el usuario lo indique.
2. El software Presentara la pantalla donde escuchara los sonidos
3. El software presentara la tablatura en pantalla

RF2: Re visualizar

1. El sistema deberá volver a graficar la última tablatura generada
2. El sistema cargara el archivo ‘save.bin’
3. El sistema graficara en la ventana las notas una vez que haya cargado las posiciones.

RF3: Salir

1. El sistema cerrara la ventana principal cada vez que el usuario lo solicita

7.1.1.1.2 Sistema

RF4: **Iniciar Micrófono**

1. El sistema estará en capacidad de utilizar el micrófono libremente.

RF5: Escuchar entorno

1. El sistema estará en capacidad de escuchar el entorno.
2. El sistema creara un archivo ‘Audio.wav’

RF6: Descomponer

1. El sistema cargara el archivo ‘Audio.wav’
2. El sistema usara la transformada rápida de Fourier sobre el archivo ‘Audio.wav’
3. El sistema llenara el vector Magnitud, Fase.
4. El sistema guardara la frecuencia hallada.

RF7: Comparar

1. El sistema tomara el valor de la variable frecuencia.
2. El sistema recorrerá los rangos de frecuencias establecidos.
3. El sistema validara que si se encuentre dentro de algunos de los rangos antes de agregarlos al vector posiciones.

RF8: Guardar

1. El sistema guardara el valor del vector posiciones en el archivo ‘save.bin’
2. En caso de no existir lo creara.

RF: Pantalla principal

1. La pantalla principal de la aplicación constara de tres botones:
    a. **Graficar:** El sistema mostrará la ventana donde se graficará la tablatura.
    b. **Re visualizar:** El sistema mostrará la ventana con las últimas posiciones guardadas
    c. **Salir:** El sistema terminara las operaciones y se cerrara.

7.1.1.2 Requisitos no funcionales

Requisitos complementarios que especifican criterios que juzgan operaciones del sistema en lugar de su comportamiento.

RNF1: Portabilidad

1. Disponibilidad de operación sobre cualquier portátil con micrófono y python.

RNF2: Interfaz y usabilidad

1. El software debe constar de una interfaz sencilla, atractiva e intuitiva.
2. La tablatura generada debe ser de fácil entendimiento.

RNF3: Rendimiento

1. Se esperan tiempos de respuesta no superiores a 2 segundos en la petición de graficación una vez se acabe el tiempo de escucha.
2. Se esperan tiempos de respuesta no superiores a 2 segundos de carga para re visualizar la tablatura.

RNF4: Precisión

1. El sistema contara con mínimo una precisión del 70%

RNF5: Tiempo de respuesta

1. El tiempo de respuesta entre descomposición de notas será de máximo 1.5 seg.
2. El tiempo de graficación del vector posición será de máximo 2 seg.

7.1.2 Modelado

En el presente apartado se describe todo el proceso de modelado del software. Se ha realizado un análisis que abarca todos los requisitos descritos en el apartado anterior. El análisis proporciona una idea completa del software desarrollado. Además, se justifican las decisiones tomadas para él.

desarrollo. El análisis del software se ha ido transformando a medida que se avanzaba el proyecto, pero se hará presente solo el diseño final en este documento.

Como se mencionó en el capítulo 4 - Metodología, se llevaron a cabo 2 modelamientos, que, aunque diferentes se complementan de la manera que se requería, a continuación, se mostraran los resultados de realizar el análisis evidenciado en los diagramas UML.

7.1.2.1 Diagrama de casos de uso

En el diagrama observamos las interacciones existentes entre el usuario y el sistema, estas son “Iniciar Micrófono” y “Re visualizar”, además se ven las interacciones que el sistema entre en sus componentes y con cierto orden de cómo funciona.

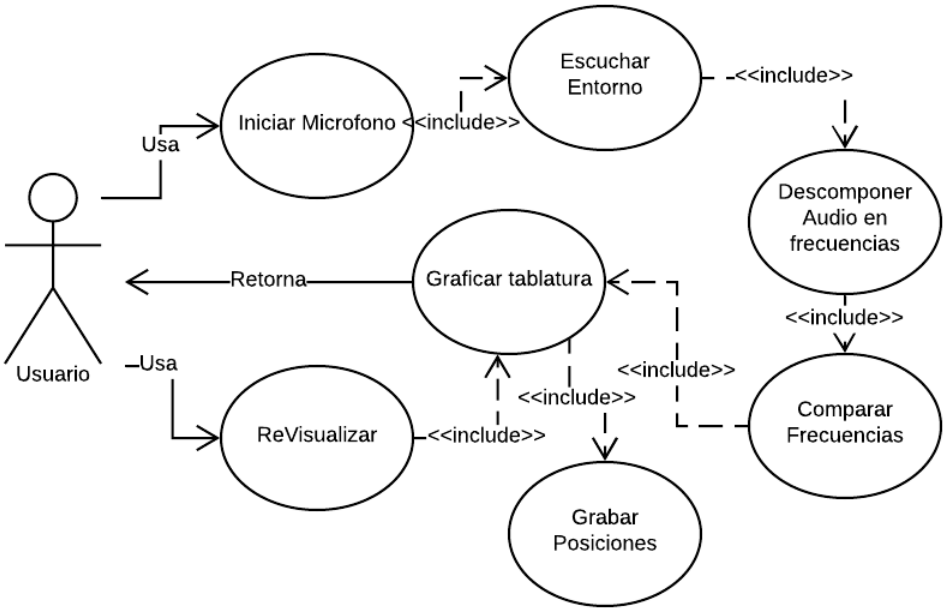

Figura 7.1: Diagrama de casos de uso.

CASO DE USO	CU1: IniciarMicrofono
Objetivo	Iniciar el uso del micrófono del equipo
Ámbito	Del sistema
Nivel	De objetivo primario
Actor Primero	Usuario
Precondiciones	Tener un micrófono disponible para su uso
Condición de Final del caso	El programa puede hacer uso libre del micrófono

CASO DE USO	CU2: EscucharEntorno
Objetivo	Escuchar y guardar en formato .wav sonidos captados por el micrófono
Ámbito	Del sistema
Nivel	De objetivo primario
Actor Primero	Sistema
Precondiciones	El programa debió activar el micrófono satisfactoriamente
Condición de Final del caso	El tiempo de escucha se agotó

CASO DE USO	CU3: DescomponerEnFrecuencias
Objetivo	Hallar las frecuencias de las notas ingresadas por el usuario mediante el uso de la transformada rápida de Fourier.
Ámbito	Del sistema
Nivel	DE objetivo primario
Actor Primero	Sistema
Precondiciones	Se debe haber guardado de forma exitosa el audio ingresado
Condición de Final del caso	Resultado exitoso del valor de frecuencia

CASO DE USO	CU4: CompararLasFrecuencias
Objetivo	El software comparará la frecuencia hallada con los rangos de frecuencia definidos y devolverá la posición de la nota si la encuentra
Ámbito	Del sistema
Nivel	De objetivo primario
Actor Primero	Sistema
Precondiciones	El sistema debió encontrar de manera satisfactoria la frecuencia del sonido ingresado
Condición de Final del caso	El sistema encontró el rango de frecuencia al que pertenece la nota.

CASO DE USO	CU5: GraficarTablatura
Objetivo	Mostrar de manera gráfica el resultado
Ámbito	Del sistema
Nivel	De objetivo primario
Actor Primero	Sistema
Precondiciones	El tiempo de escucha se agotó
Condición de Final del caso	Gráfica generada satisfactoriamente

CASO DE USO	CU6: GrabarPosiciones
Objetivo	El sistema generara un archivo 'save.bin' donde almacenara las posiciones de la última tablatura graficada
Ámbito	Del sistema
Nivel	De objetivo secundario
Actor Primero	Sistema
Precondiciones	Haber graficado una tablatura satisfactoriamente
Condición de Final del caso	Guardar las posiciones de manera exitosa

CASO DE USO	CU7 Re-VisualizarTablatura
Objetivo	El sistema cargara las posiciones del archivo 'save.bin' para volver a generar la última tablatura graficada
Ámbito	Del sistema
Nivel	De ámbito secundario
Actor Primero	Usuario
Precondiciones	Haber guardado las posiciones en el archivo satisfactoriamente
Condición de Final del caso	Gráfica Re generada satisfactoriamente

Tabla 7.1: Casos de uso.

7.1.2.2 Diagrama de clases

El diagrama de clases nos enseña de una manera más detalla el número de clases, los métodos y atributos de las mismas y como se comunican entre ellas.

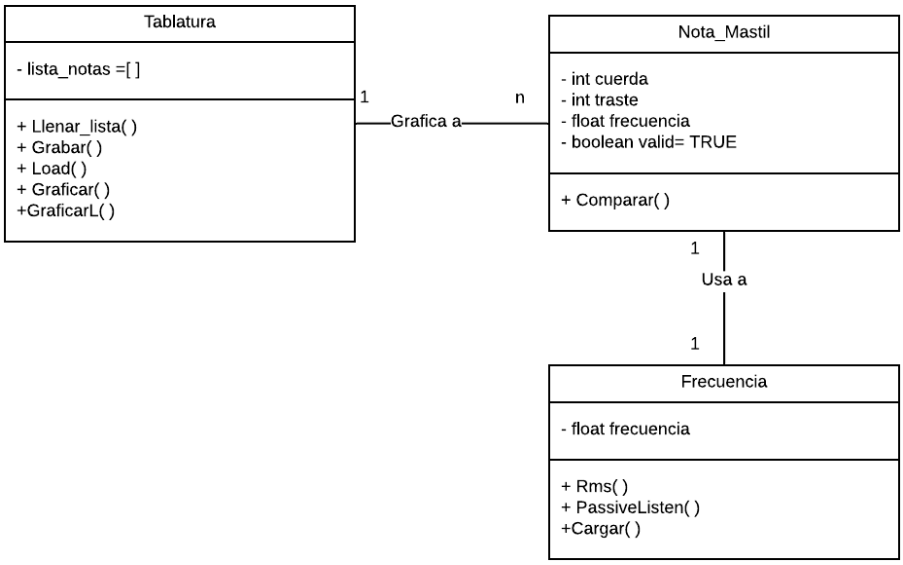

Figura 7.2: Diagrama de clases

7.1.2.3 Diagrama Entidad-Relación

El diagrama entidad relación nos presenta de una manera simplicada lo que se plasma en el diagrama de clases, mostrándonos solo los atributos y los atributos clave, es bastante útil para ingresar los campos directamente en una base de datos.

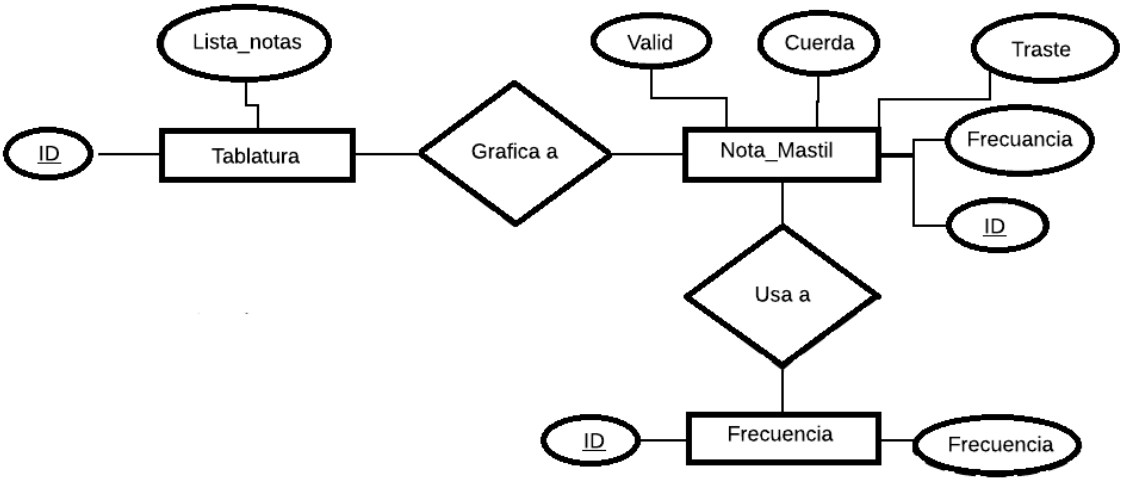

Figura 7.3: Diagrama Entidad-Relación

7.2 Diseño

7.2.1 Arquitectura del software

Como se menciona en el capítulo 4 - Metodología la arquitectura del software se basa en el Modelo- Vista- Controlador, a continuación, se explica mejor el funcionamiento de esta arquitectura y como se implementó.

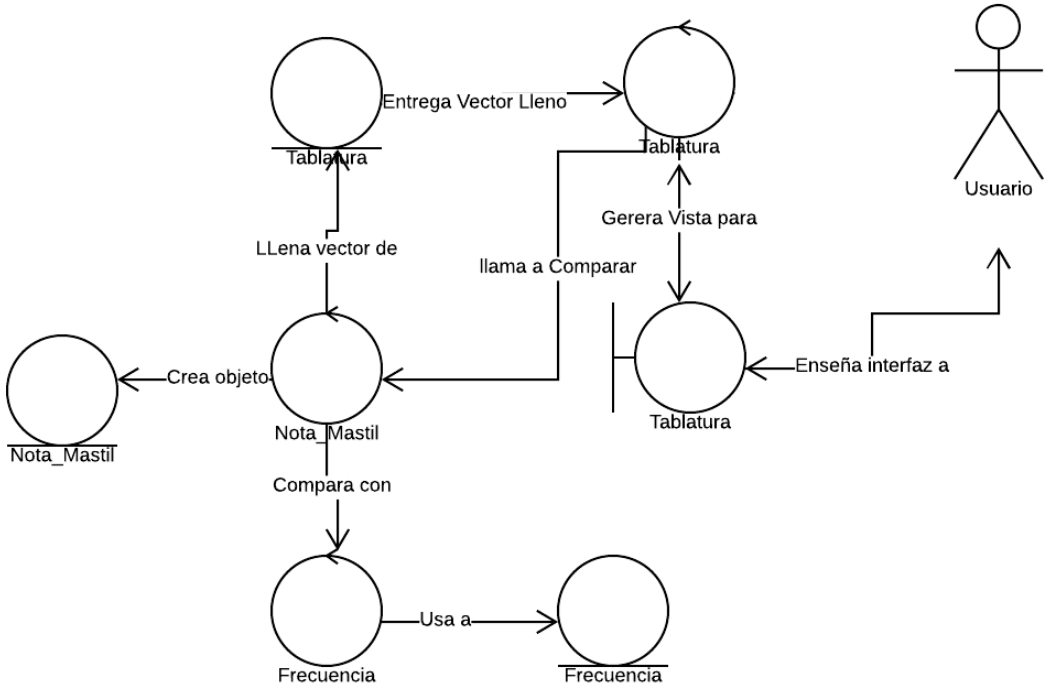

Figura 7.4: Modelo – Vista - Controlador

En el diagrama anterior apreciamos que el usuario se comunica con el sistema únicamente mediante el Control de tablatura, el cual a su vez se comunica con los demás modelos de manera ordenada, para obtener los datos necesarios, una vez que esta comunicación es satisfactoria y se acabe el tiempo de escucha, el Modelo tablatura le indica a la Vista tablatura que enseñe sus resultados al usuario mediante una interfaz gráfica.

7.2.2 Diagrama de despliegue

El diagrama de despliegue nos muestra el "nodo" con el que el usuario va a interactuar, en este caso el Pc, y sus componentes internos necesarios para la ejecución, en este caso el "nodo" requiere de python, y ciertas librerías descritas a continuación.

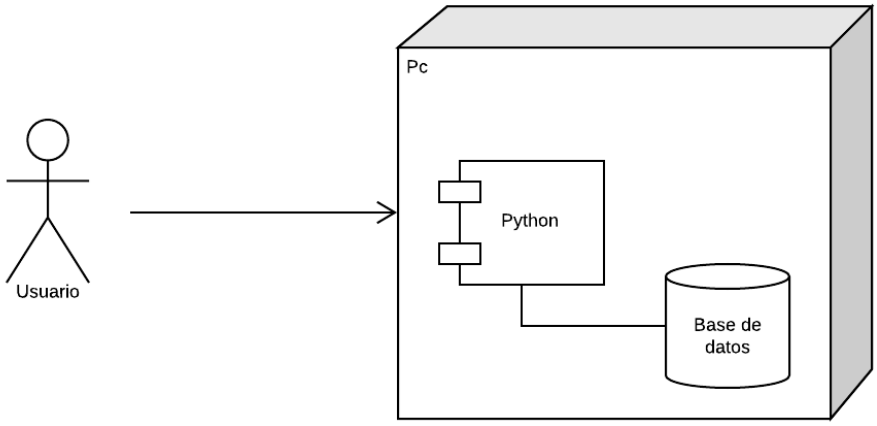

Figura 7.5: Diagrama de despliegue.

Librería	Uso
Math	Operaciones matemáticas simples
Pygame	Entorno grafico de python
Seaborn	Representación de datos gráficos
librosa	Análisis y manejo de audio
numpy	Funciones matemáticas de alto nivel
Pyaudio	Grabar, reproducir y transmitir audio
estruct	Manejo de estructuras de datos
wave	Manejo de archivos .wav
Scipy	Funciones matemáticas de alto nivel
Os	Funciones del sistema
Tkinter	Manejo de ventanas
pickle	Creación y manipulación de archivos

Tabla 7.2: Librerías necesarias dentro de python.

7.2.3 Diagramas de actividades

El diagrama de actividad a continuación es un desglose del método de la clase Frecuencia, PassiveListen (). Esta se encarga de cargar el audio almacenado en 'Audio.wav' y aplica la transformada rápida de Fourier para guardar los valores de magnitud y fase y retornar la frecuencia hallada en base a estos valores.

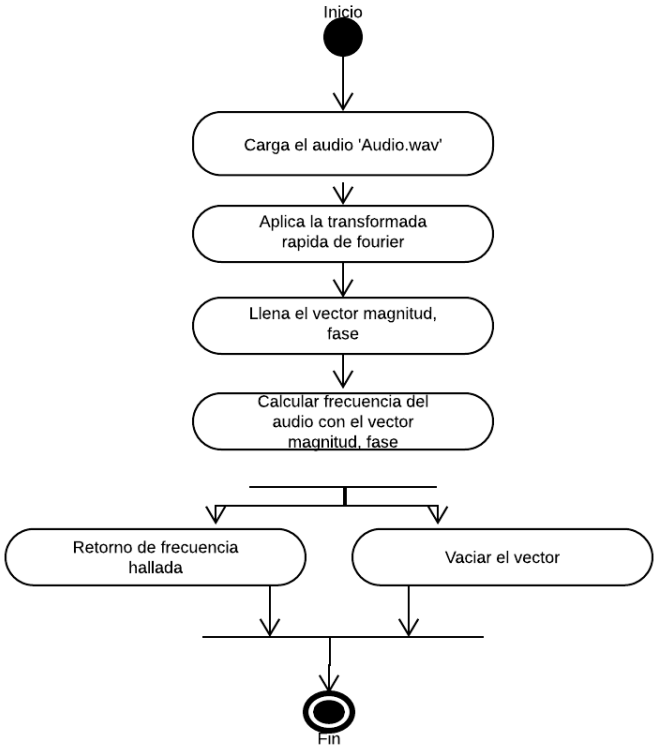

Figura 7.6: Diagrama de actividad, caso de uso PassiveListen

El diagrama de despliegue a continuación es un desglose del método de la clase Nota_Mastil, Comparar (). Este se encarga de utilizar el valor de la frecuencia de la clase Frecuencia para compararla con los rangos de frecuencia definidos, esta se asegura de que la

frecuencia no sea menor que 70 ni mayor a 296, y a su vez de que la frecuencia si está en algunos de los rangos, si no cumple alguna de las condiciones la frecuencia es descartada y se vuelve a escuchar, si la frecuencia si se encuentra entre los rangos las posiciones de la nota se agregan a un vector de posiciones, el método realizara este proceso hasta que el tiempo acabe.

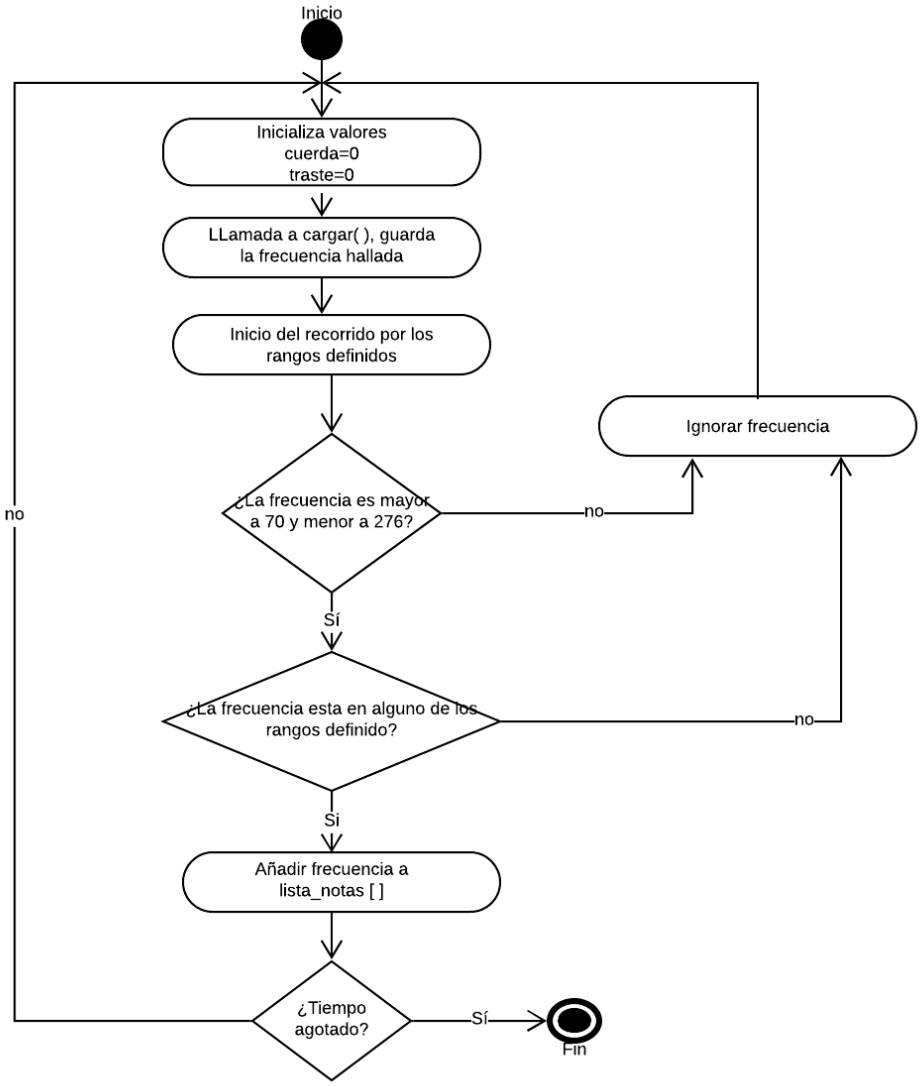

Figura 7.7 Diagrama de actividad, caso de uso Comparar

El siguiente diagrama es un desglose del método de la clase Tablatura, Graficar. esta se encarga de utilizar el vector de posiciones Lista_notas [ ] para graficarlas en la posición correspondiente en

la tablatura y guardarlas en el archivo ‘save.bin’. además, es quien presenta los resultados por pantalla al usuario.

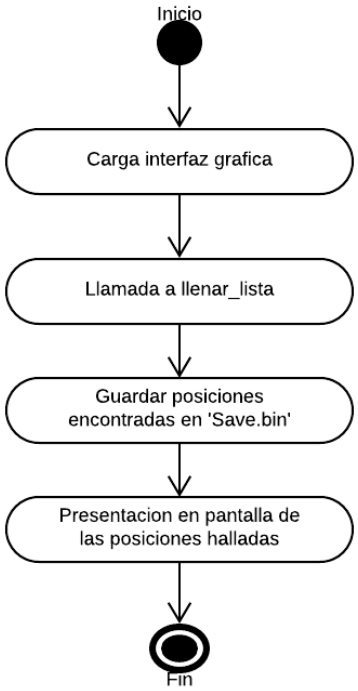

Figura 7.8: Diagrama de actividad. Caso de uso Graficar

El siguiente diagrama es un desglose del método de la clase Tablatura, GraficarL. Esta se encarga de cargar el vector de posiciones Lista_notas directamente del archivo ‘save.bin’ y re graficara las posiciones para posteriormente volver a mostrar la última tablatura guardada en pantalla.

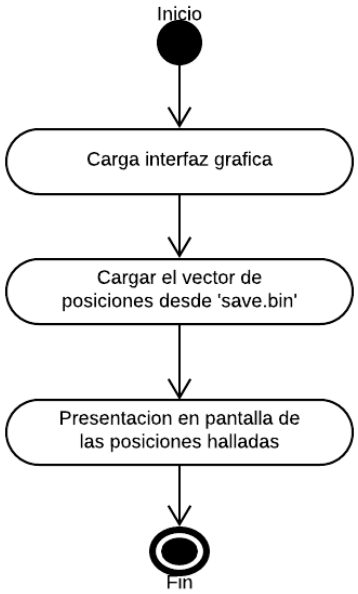

Figura 7.9: Diagrama de actividad, caso de uso GraficarL

7.3 Tecnologías utilizadas

Como se afirma en el numeral 7.2.2 Diagrama de despliegue la tecnología principal y bajo el cual se construyó la totalidad del proyecto es el lenguaje de programación Python. Ya que su manejo de audio y de las señales que lo acompañan es considerablemente más preciso que en otros lenguajes de programación conocidos por el desarrollador, ya que python cuenta con librerías especialmente diseñadas para el manejo de audio dando la posibilidad de grabar en nuestro ordenador un archivo con el audio que ha escuchado, además de conceder la posibilidad de modelar los objetos a nuestro gusto e inicializarlos de manera sencilla bajo los parámetros que establezcamos, pudiendo llenar después estructuras de datos como vectores con estos objetos, algo bastante útil para la implementación del sistema. Para este proyecto se ha utilizado específicamente la versión 2.7 de python, obtenido desde portal oficial de python.

Para la realización de los diagramas de modelado UML se utilizó una herramienta online gratuita que permite la creación de todos estos modelos de manera fácil y rápida, al ser intuitiva, esta es Lucidchart.

7.4 Diseño interfaz de usuario

Para la implementación gráfica del sistema se realizaron primero algunos diseños en papel, para asegurarse de que la interfaz era amigable y era lo que el usuario necesitaba, después se plasmaron los ajustes necesarios en mockups. Estos mockups son una herramienta de diseño cuya intención es mostrar de manera sencilla y sin funcionalidades (aun) de como lucirá el sistema una vez finalizado su desarrollo. A continuación, veremos los mockups diseñados para el sistema.

Figura 7.10: Mockup inicio.

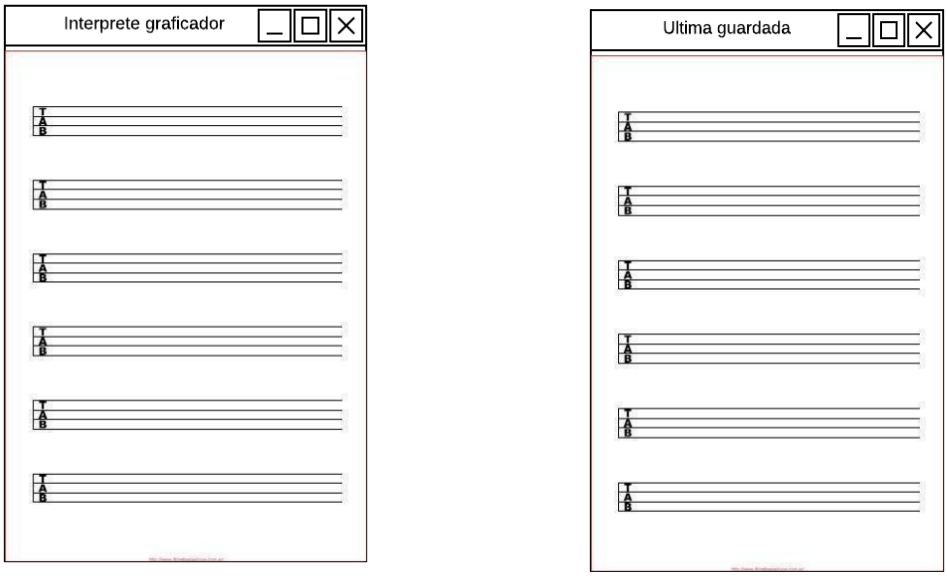

Figura 7.11: Mockups para el graficado

7.5 Implementación

En este apartado se abordan las fases de implementación del proyecto. Se explicarán algunos detalles relevantes, así como las diferentes dificultades que se encontraron en el proceso y las pruebas implementadas en el mismo.

En esta fase se implementa el diseño realizado en el apartado anterior de diseño, basado en el apartado 7.1.1 – catálogo de requisitos, Como se ha especificado en este apartado se utilizará el lenguaje de programación python, junto a todas las librerías detalladas anteriormente, basando el diseño de la implementación en los diagramas UML del capítulo apartado anterior.

7.5.1 Desarrollo del proyecto

Al tener ya claro lo que se requería para el sistema, cuantas partes tendría, y como se comunicarían gracias al análisis y al diseño previo, la implementación del sistema no resulto ser tan complicada como se esperaba, primero definimos entonces las clases a usar.

```
class Frecuencia:
class Nota_Mastil:
class tablatura:
```

Figura 7.12: Clases

Se necesitaba empezar a implementar el apartado de la clase **Frecuencia** ya que esta quien escucha, graba, y descompone.

El diseño de esta clase nos permitirá empezar a conocer los rangos de frecuencia se mueven las notas generadas por el bajo eléctrico y se necesitaba esta información para continuar con los demás apartados, por eso fue la primera en implementarse, a continuación, veremos los métodos que la componen.

```
def Rms(self,frame):
    count = len(frame)/2
    format = "%dh"%(count)
    shorts = struct.unpack( format, frame )
    sum_squares = 0.0
    for sample in shorts:
        n = sample * (1.0/32768.0)
        sum_squares += n*n
    rms = math.pow(sum_squares/count,0.5);
    return rms * 1000
```

Figura 7.13: Método Rms

```
def PassiveListen(self,audio):
    CHUNK = 1024; RATE = 8000; THRESHOLD = 200; LISTEN_TIME = 20
    didDetect = False
    # prepare recording stream
    p = pyaudio.PyAudio()
    stream = p.open(format=pyaudio.paInt16, channels=1, rate=RATE, input=True, frames_per_buffer=CHUNK)
    # stores the audio data
    all =[]
    # starts passive listening for disturbances
    #print RATE / CHUNK * LISTEN_TIME
    for i in range(0, RATE / CHUNK * LISTEN_TIME):
        input = stream.read(CHUNK)
        rms_value = self.Rms(input)
        #print rms_value
        if (rms_value < THRESHOLD):
            didDetect = True
            #print "Listening...\n"
            break
    if not didDetect:
        stream.stop_stream()
        stream.close()
        return False
    # append all the chunks
    all.append(input)
    for i in range(0, 7):
        data = stream.read(CHUNK)
        all.append(data)
```

```
    # save the audio data
    data = ''.join(all)
    stream.stop_stream()
    stream.close()
    wf = wave.open('Audio.wav', 'wb')
    wf.setnchannels(1)
    wf.setsampwidth(p.get_sample_size(pyaudio.paInt16))
    wf.setframerate(RATE)
    wf.writeframes(data)
    wf.close()
    return True
```

```
def Cargar(self):
      mic = Frecuencia()
      a=mic.PassiveListen('ok Google')
      x, fs = librosa.load('Audio.wav', sr=44100)
      f = numpy.linspace(0, fs, 8192)
      X = scipy.fft(x[:8192])
      X_mag = numpy.absolute(X)
      temp = numpy.argpartition(-X_mag, 10)
      result_args = temp[:10]
      result_args.sort()
      plt.plot(f[:1000], X_mag[:1000]) # magnitude spectrum
      plt.xlabel('Frequency (Hz)')
      FrecNota=f[result_args[0]]
      print FrecNota
      return FrecNota
```

Figura 7.14: Método Passivelisten y cargar

Como se apreció en la imagen anterior la implementación está acorde a lo que se definió en los diagramas de clases del capítulo 4- Diseño. Estos se encargan de escuchar guardar y aplicar la transformada de Fourier respectivamente, El diagrama de actividades del capítulo 4 nos enseña de una manera más amplia su funcionamiento.

Una vez implementada la clase que se encargara de escuchar y descomponer se hicieron múltiples pruebas con el bajo eléctrico, se anotaron las frecuencias que más resaltaban al generar una nota y se guardaron aparte, estas frecuencias fueron corroboradas en múltiples ocasiones para evitar usar datos erróneos, y a partir de estas se diseñaron los rangos bajo los cuales se moverán los resultados.

```
def __init__(self,frecuencia):
    self.cuerda=0
    self.traste=0
    self.frec=frecuencia
    self.valid=False

def Comparar(self):
    frec=int(Frecuencia().Cargar())
    #print frec
    if frec>(70) and frec<(276):
        if frec>=(80) and frec<=(82):
            self.cuerda= 4
            self.traste =0
            self.valid=True

        if frec>=(105) and frec<=(109):
                self.cuerda=3
                self.traste=0

        if frec>=(137) and frec<=(141):
                self.cuerda=2
                self.traste=0
                self.valid=True
```

Figura 7.15 Método Comparar y rangos de frecuencia

Al terminar este apartado, que hace parte de la clase **Nota_Mastil** se implementó en totalidad la clase, al contar ya con los rangos de frecuencia solo fue necesario definir el método que iba a ser llamado por la clase **Tablatura**, este sería Comparar, quien utilizando el atributo de la clase **Frecuencia** recorrería los rangos definidos, una vez encontrado el rango al que pertenece se crea un objeto **Nota_Mastil** y se le asignan los atributos cuerda y traste dependiendo del rango que haya encontrado. Para evitar recorridos innecesarios se ha definido que el valor mínimo de entrada debe ser mayor a 70 Hz y máximo de 276 Hz, así se asegura que la frecuencia ingresada podría pertenecer a una nota generada por un bajo eléctrico. Además, el sistema se asegura que no haya datos vacíos, guardando dentro del vector lista_notas [ ] únicamente los objetos que tienen asignado algún valor de posición.

```
def Graficar1(self):
    running=True
    while True:
        for eventos in pygame.event.get():
            if eventos.type == pygame.QUIT:
                exit()
        if running:
            pantalla=pygame.display.set_mode([500,708])
            pygame.display.set_caption("Interprete Graficador")
            fondo=pygame.image.load('Blanco.jpg')
            pantalla.blit(fondo,(0,0))
            pygame.display.update()
            self.Llenar_lista()
            self.Grabar()
            lista_graficar=self.lista_notas
            pygame.display.update()
            j=0
            k=0
            for e in self.lista_notas:
                pantalla.blit(Numero2.render(str(e.traste),True,[254,127,156]),[70 + (j*35),42+(e.cuerda*14)+ (k*102)])
                j+=1
                if j==10:
                    j=0
                    k+=1
                pygame.display.update()
            running=False
```

Figura 7.16: Método Graficar

Ya teniendo la parte gruesa del sistema solo nos resta realizar la parte gráfica y el llamado a las demás funciones que hacen parte de la clase **Tablatura**, teniendo los mockups no se tuvieron muchos inconvenientes. Para la venta principal se utilizó la librería Tkinter de python, ya que el manejo de las ventanas es más sencillo con ella, diseñamos la ventana principal y el primer llamado a graficar. Para esta parte se decidió que creara una ventana con la librería Pygame, dado que se graficara sobre una imagen las posiciones almacenadas en el vector lista_notas [ ] y es más sencillo con esta librería, a su vez el vector posición es guardado en un archivo 'save.bin' en nuestra máquina, permitiendo graficar nuevamente estas posiciones más adelante.

Para ver con más detalle el funcionamiento de estos métodos críticos en el capítulo 4 – Diseño se muestran los diagramas de diseño del sistema.

```
def Grabar(self):
    with open('save.bin', 'w') as archivo:
        pickle.dump(self.lista_notas,archivo)

def Load(self):
    with open("save.bin","r") as archivo:
        notas=pickle.load(archivo)
        return notas
```

Figura 7.17: Método Grabar y Load

Una vez implementado todo por aparte, solo resta juntar y hacer los llamados a las funciones definidas. Lo primero que se le presentará al usuario será la ventana principal de Tkinter, en ella habrán 2 opciones de graficado, la primera genera el vector lista_notas [ ] vacío y llama a su función comparar para llenarlo, esta a su vez hereda el valor de frecuencia de la clase **Frecuencia** para hacer las debidas comparaciones, entrando en un bucle de escucha, descomposición, comparación, guardado hasta que el tiempo se termine, una vez que esto suceda el método Graficar mostrara los valores almacenados dentro del vector lista_notas [ ], y creando el archivo ´save.bin´. La opción Re visualizar, carga directamente el vector lista_notas [ ] del archivo 'save.bin' y re grafica las notas de la misma manera que lo hace la opción 1.

```
def __init__(self):
    self.lista_notas=[]

def Llenar_lista(self):
    i=0
    limite=30
    while i<limite:
        f_n=Frecuencia()
        f1=f_n.Cargar()
        f=Nota_Mastil(f1)
        f.Comparar()
        if f.valid:
            self.lista_notas.append(f)
        i+=1
```

Figura 7.18: Método Llenar_lista

```
def GraficarL(self):
    while True:
        for eventos in pygame.event.get():
            if eventos.type == pygame.QUIT:
                exit()
        pantalla=pygame.display.set_mode([500,708])
        pygame.display.set_caption("ultima guardada")
        fondo=pygame.image.load('Blanco.jpg')
        pantalla.blit(fondo,(0,0))
        lista_graficar=self.Load()
        pygame.display.update()
        j=0
        k=0
        for e in lista_graficar:
            pantalla.blit(Numero2.render(str(e.traste),True,[254,127,156]),[70 + (j*35),42+(e.cuerda*14)+ (k*102)])
            j+=1
            if j==10:
                j=0
                k+=1
        pygame.display.update()
```

Figura 7.19: Método GraficarL

7.1.1 Pruebas

Se realizaron una serie de pruebas una vez se realizó el desarrollo, para asegurarse de que el sistema cumplía en cabalidad con los requisitos los resultados se plantean a continuación.

Prueba del software Interprete graficador			
ID/Nombre Caso de Prueba: CPA01	Autor del Caso de Prueba: Alejandro Plaza Parra		
Versión del Caso de Prueba: 1	Fecha de Creación: 10/07/19		
ID Caso de Uso: RF-1	Fecha de Ejecución: 16/07/19		Versión: 1.0
Flujo de pasos de la Prueba:			
Objetivo de la Prueba			
El sistema graficara unicamente cuando el usuario lo indique			
Nro.	**Descripción del Escenario**	**Resultado Esperado**	**Resultado Real**
1	Graficar la tablatura	El sistema esperará las acciones del usuario para ejecutarse	El sistema espero correctamente las acciones del usuario
paso	***Paso a paso del escenario de prueba***		***Observaciones generales durante la prueba***
1	Ejecutar el software.		
2	Esperar un tiempo (5mins)		
3	Ejecutar alguna accion en la interfaz		
Decisión de Aprobación del Caso de Prueba: Aprobó: __x__ Fallo: ____			
Nombre y firma del Probador	Alejandro Plaza Parra		

Tabla:7.3 Prueba de Requisito funcional 1.

Prueba del software Interprete graficador			
ID/Nombre Caso de Prueba: CPA02	Autor del Caso de Prueba: Alejandro Plaza Parra		
Versión del Caso de Prueba: 1	Fecha de Creación: 10/07/19		
ID Caso de Uso: RF-5	Fecha de Ejecución: 16/07/19		Versión: 1.0
Flujo de pasos de la Prueba:			
Objetivo de la Prueba			
El sistema escuchara, guardara y descompondra el audio escuchado			
Nro.	**Descripción del Escenario**	**Resultado Esperado**	**Resultado Real**
1	Crear el archivo "Audio.wav' y encontrar el valor de la frecuencia	El sistema tendra el valor de la frecuencia en una variable despues de escuchar el entorno	El sistema encontró y asigno el valor de la frecuencia a la variable satisfactoriamente
paso	***Paso a paso del escenario de prueba***		***Observaciones generales durante la prueba***
1	Ejecutar el software.		
2	Oprimir el boton grabar		
3	Esperar la ejecución del sistema		
4	Revisar que el archivo'save.bin' se encuentre en la carpeta del proyecto		
5	Revisar elel contenido del archivo 'save.bin'		
6	Revisar el contenido de la variable frecuencia		
Decisión de Aprobación del Caso de Prueba: Aprobó: __x__ Fallo: ____			
Nombre y firma del Probador		Alejandro Plaza Parra	

Tabla:7.4 Prueba de Requisito funcional 5.

Prueba del software Interprete graficador			
ID/Nombre Caso de Prueba: CPA03	Autor del Caso de Prueba: Alejandro Plaza Parra		
Versión del Caso de Prueba: 1	Fecha de Creación: 10/07/19		
ID Caso de Uso: RF-7	Fecha de Ejecución: 16/07/19		Versión: 1.0
Flujo de pasos de la Prueba:			
Objetivo de la Prueba			
El sistema comparará y asignara el valor de las posiciones al vector Lista_notas [ ]			
Nro.	**Descripción del Escenario**	**Resultado Esperado**	**Resultado Real**
1	El sistema guardara las posiciones encontradas de las frecuencias	El vector Lista_notas [ ] tendra guardados valores de posiciones	El vector Lista_notas [ ] tiene las posiciones guardadas
paso	***Paso a paso del escenario de prueba***		***Observaciones generales durante la prueba***
1	Ejecutar el software.		
2	Oprimir el boton grabar		
3	Esperar la ejecución del sistema		
4	Revisar el contenido del vector Lista_notas [ ]		
Decisión de Aprobación del Caso de Prueba: Aprobó: __x__ Fallo: ____			
Nombre y firma del Probador		Alejandro Plaza Parra	

Tabla:7.5 Prueba de Requisito funcional 7

Prueba del software Interprete graficador			
ID/Nombre Caso de Prueba: CPA04	Autor del Caso de Prueba: Alejandro Plaza Parra		
Versión del Caso de Prueba: 1	Fecha de Creación: 10/07/19		
ID Caso de Uso: RF-8	Fecha de Ejecución: 16/07/19		Versión: 1.0
Flujo de pasos de la Prueba:			
Objetivo de la Prueba			
El sistema guardara el valor Lista_notas en el archivo 'save.bin'			
Nro.	**Descripción del Escenario**	**Resultado Esperado**	**Resultado Real**
1	El sistema guardara el vector Lista_notas en el archivo 'save.bin' si este no existe lo creara	save.bin' tendra los valores de Lista_notas	El sistema creo el archivo en la carpeta del proyecto y guardo satisfactoriamente los valores
paso	***Paso a paso del escenario de prueba***		***Observaciones generales durante la prueba***
1	Ejecutar el software.		
2	Oprimir el boton grabar		
3	Esperar la ejecución del sistema		
4	Revisar que el archivo 'save.bin' se encuentre en la carpeta del proyecto		
5	Revisar que el contenido del archivo 'save.bin'		
Decisión de Aprobación del Caso de Prueba: Aprobó: __x__ Fallo: ____			
Nombre y firma del Probador	Alejandro Plaza Parra		

Tabla:7.6 Prueba Requisitos funcional 8

8

Conclusiones y dificultades

Este capítulo se dedica a conclusiones finales y personales del proyecto, extraídas a lo largo de todo el proceso de desarrollo del mismo.

Este apartado se redacta a modo de resumen final del capítulo con el fin de condensar los aspectos más importantes y algunas de las dificultades encontradas.

Se han desarrollado con éxito los módulos previstos en la fase de diseño, las interfaces y las tareas de cada clase que completan la funcionalidad del software

El software desarrollado ofrece la comodidad de diseñar la tablatura por el usuario, además de soportar memoria para re visualizar la última tablatura que fue generada.

Dado el conocimiento adquirido durante los años en la carrera, tanto es asignaturas directamente ligadas al lenguaje como las que se centran más en desarrollo de software, no han surgido problemas que no hayan podido solucionar haciendo uso de internet y de documentación de las librerías usadas.

El problema más remarcable que se encontró en la implementación radicaba en cómo mostrar la tablatura generada, ya que al comienzo solo se planeaba usar Tkinter el cual no contiene un entorno grafico muy poderoso, limitando mucho lo que se podía hacer por lo que se estaban complicando las cosas sin necesidad, al final se optó por usar Pygame junto a Tkinter para que la interfaz fuera como se deseaba, dejando la ventana principal en Tkinter con su capacidad para manejar demás ventanas y el apartado gráfico solo a Pygame, usando este conjunto se logró lo que

se buscaba, una comunicación entre ambas que posibilitara el cumplimiento de los objetivos del proyecto.

Algunos problemas recayeron en la llenada del vector lista_notas [ ] ya que en diseño original cuando no encontraba un rango de frecuencias al cual perteneciera la nota ingresada (ruido), agregaba un objeto con valores =0, ocasionando que en la gráfica se visualizaran 0 que no deberían existir. Esto se solucionó agregando una variable boolean para que el sistema asegurara que si había entrado a un rango antes de añadirla al vector.

Además de haber aprendido a desarrollar un proyecto de principio a fin, pasando por todas sus fases, se adquirieron nuevos conocimientos, o ampliado muchos de ellos, se aprendió a utilizar correctamente muchas librerías de las que dispone python, además de ampliar el conocimiento sobre manejo de ventanas en el mismo, y otros necesarios que se han requerido durante todas las fases, resaltando sobre todo el modelamiento UML donde tuve que trabajar un poco más ya que era muy abstracto.

Adicional se les dan conclusiones respecto a los objetivos específicos del proyecto, dando final al desarrollo y a la implementación

- Se logró el uso de librerías necesarias para manejo de audio.
- Se realizó el análisis y almacenamiento de los sonidos emitidos.
- Se hizo uso de la transformada rápida de Fourier.
- Se realizó la Comparación de frecuencias con los rangos definidos.
- Se logró el almacenamiento de las posiciones encontradas en las comparaciones.
- Se desarrolló la graficación y guardado del archivo de posiciones
- Se logró el Desarrollo de la función para re visualiza

9

Recomendaciones

Dentro del trabajo futuro de este proyecto se pueden destacar varias mejoras.

La primera de ellas es mejorar la interfaz gráfica, con ayuda de una persona especialista se puede lograr una aplicación con un acabado más profesional.

Puede ampliarse el trabajo realizado, haciendo ajustes para que la cantidad de bajos que pueden ser partícipes del software sea cada vez mayor, incluyendo bajos de todos los tipos de cuerdas existentes, ampliando el número de cuerdas, pudiendo reconocer las notas a pesar diferentes técnicas que se utilicen.

Se puede ampliar la gama de instrumentos que pueden hacer uso de este sistema, incluyendo otros instrumentos de cuerdas como guitarra, contrabajo, ukulele etc.

Con un poco más de trabajo sobre las señales se podrían introducir piezas musicales completas y que el sistema sea capaz de graficar la tablatura del instrumento que se le solicite, ignorando los demás.

La velocidad del reconocimiento de notas puede ser aumentada, pudiendo ingresar más notas por segundo.

La precisión puede ser aumentada considerablemente si se añaden más valores a tener en cuenta a la hora de calcular la frecuencia.

REFERENCIAS

[1] CORTES, Jimmy. KNOTT, Andrew. CHAVES, José. Aproximación a la síntesis de la música a través del análisis de Fourier. Redalyc. [en línea], [Revisado 17 de julio de 2019]. Disponible en internet: https://www.redalyc.org/html/849/84925149018/

[2] BERNAL, Jesús. GOMEZ, Pedro. BOBADILLA, Jesús. Una visión práctica en el uso de la transformada de Fourier como herramienta para el análisis espectral de la voz. Ub. [en línea], [Revisado 17 de julio de 2019]. Disponible en internet: stel.ub.edu/labfon/sites/default/files/EFE-X-JBernal_PGomez_JBobadilla FFT_una_vision_practica_herramienta_para_el_analisis_espectral_de_la_voz.pdf

[3] E Oran Brigham, The Fast Fourier Transform and it Aplication [en línea], [Revisado 17 de julio de 2019]. Disponible en internet: http://sar.kangwon.ac.kr/gisg/FFT_book.pdf

[4] Jaramillo Ana María, “Acústica la ciencia del sonido”, editorial ITM; (enero 1, 2000)

Anexo A

Manual de usuario

Al arrancar el software se mostrará una pantalla de inicio.

Figura A1: Pantalla de inicio

El software esperara en esta ventana hasta que el usuario presione alguno de los botones

El software empezara a escuchar una vez el botón sea pulsado y mostrara la ventana en blanco hasta que el tiempo de escucha.

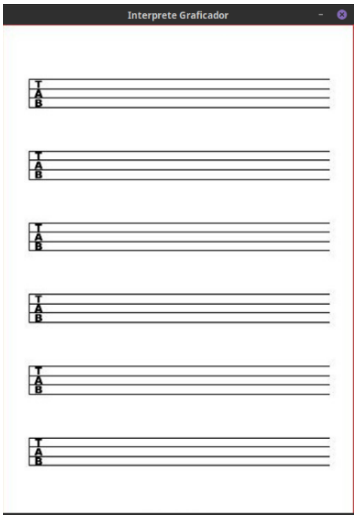

Figura A2: Ventana al presionar “graficar”

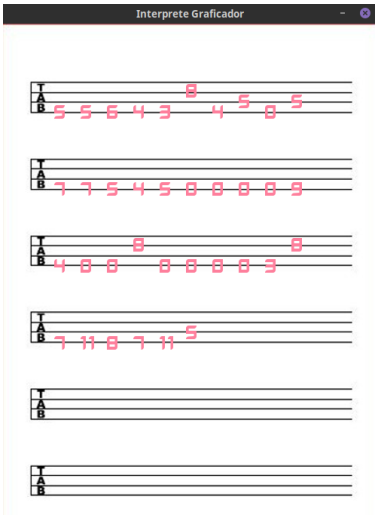

Figura A3: Resultado “graficar”

Una vez el tiempo se agote se mostrará en pantalla el resultado de las notas que el usuario ingreso.

Botón “Re visualizar”

Si el software no ha sido ejecutado anteriormente el software mostrara una tablatura vacía, ya que el archivo ‘save.bin’ no existe en ese momento.

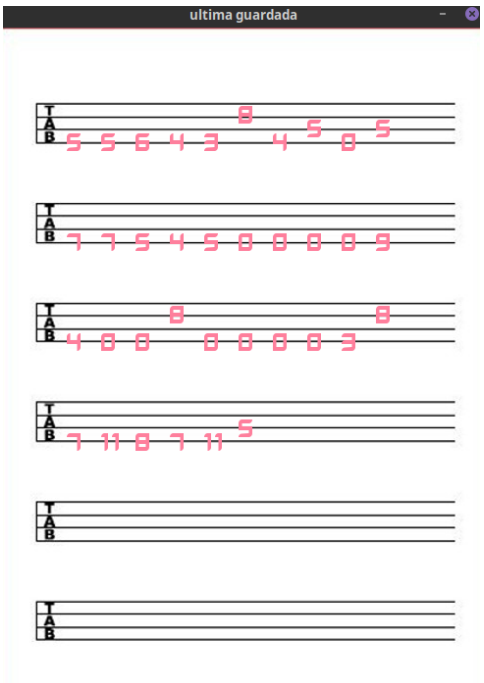

Figura A4: Re visualizar

Cuando el programa sea ejecutado y se guarden las posiciones, el software mostrara con “Re visualizar” estas últimas posiciones guardadas en ‘save.bin’

Anexo B

El código final, con todos sus componentes estarán disponibles de forma pública para su descarga en el siguiente enlace.

https://github.com/ARK3540/PoyectoGrado/tree/master/GraficadorTabs